國立高雄科技大學 109 學年度碩士班 招生考試　試題紙

系 所 別：電機工程系碩士班　　　　組　別：丙組

考科代碼：2015　　　　考　科：資料結構

＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝＝

注意事項：

1、各考科一律可使用本校提供之電子計算器，**考生不得使用自備計算器**，違者該科不予計分。

2、請於答案卷上規定之範圍作答，違者該題不予計分。

3、本試題含單選題共 5 大題，共 100 分。

4、考生作答前請詳閱答案卷之考生注意事項。

1. Multiple choice question. **Only one answer is correct**. (20%)

(1) If we have the following declaration: **int i, *pi;**
Which code can assign the address of **i** to **pi**.
- (A) pi = i;
- (B) i = *pi;
- (C) pi = *i;
- (D) pi = &i;

(2) Which of the following description about arrays (in C) is an **incorrect** description.
- (A) **int list[5];** declares an array consists of five integers.
- (B) **int *list[5];** declares an array consists of five pointers to integers.
- (C) **int **myArray;** declares a pointer that can point to a two-dimensional array.
- (D) Arrays in C are indexed starting at 1.

(3) Which of the following description about stacks and queues is an **incorrect** description.
- (A) A stack is an ordered list in which insertions and deletions are made at the bottom element.
- (B) A queue is an ordered list in which insertions and deletions take place at different ends.
- (C) A stack is also known as a last-in-first-out list.
- (D) A queue is also known as a first-in-first-out list.

(4) Which of the following description about expressions is **incorrect** (C precedence).
- (A) The postfix notation of ***a*/*b*-*c*+*d***e*-*a***c*** is ***ab*/*cde***ac**-+-**.
- (B) ***a***b*/*c*** is an infix notation.
- (C) The postfix notation of ***a***b*/*c*/-(*e*+*f*)** is ***ab***c*/*ef*+-/**
- (D) The prefix notation of ***a**(*b*+*c*)/*d*-*g*** is **-/**a*+*bcdg***

(5) How many times are the underline statement in the following code executed?

```
for (a = 1; a < m; a++)
    for(b = 1; b <= n; n++)
        c++;
```

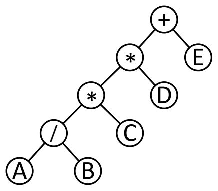
(A) n*(m-1)

(B) (n+1)*m

(C) n*m

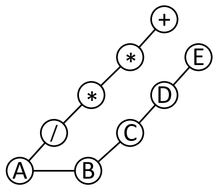
(D) m+n

(6) Which of the following description about the depicted tree is **incorrect**.



(A) The tree is a binary tree.

(B) The left child-right sibling representation of the tree is:



(C) The list representation of the tree is: **(+(*(*(/(A, B), C), D), E))**

(D) The post-order traversal of the tree gets: **AB/C*D*E+**

(7) Which sorting method has the **worst** average time complexity?

(A) Insertion sort

(B) Heap sort

(C) Merge sort

(D) Quick sort

Assume that you want to sort a list of numbers in ascending order. The candidate methods include: (A) Count sort; (B) Insertion sort; (C) Internal sort; (D) Quick sort; (E) Merge sort; (F) Heap sort; (G) Bubble sort; (H) Selection sort; (I) Radix sort; (J) Table sort. Please choose an appropriate sorting method from the above candidates for each of the following descriptions:

(8) From the unsorted list, find the smallest one and place it next in the sorted list.

(9) Perform several left-to-right passes over the unsorted list. In each pass, pairs of adjacent elements are compared and exchanged if necessary. Terminate the sorting if no elements are exchanged.

(10) Select a reference element among the elements in the list to be sorted. The elements are then reordered so that the values of the elements to the left of the reference are less than or equal to that of the element and those of the elements to the right of the reference are greater than or equal to that of the reference. Finally, the elements to the left of the reference and those to its right are sorted recursively.

2. Use the graph shown in Figure 1 to answer the following questions. (10%)
   (1) Assume start from **node 5**. Show the results of the breadth first traversal. (5%)
   (2) Assume start from **node 3**. Show the results of the depth first traversal. (5%)

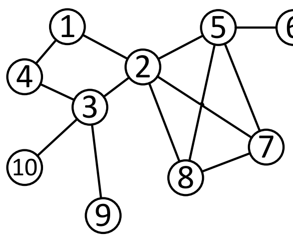


Figure 1. Graph

3. As shown in Figure 2, a hash table has been inserted with six keys in a sequence by using a hash function ***h*(*k*) = *k* % 10**, and **the liner probing** method to resolve collisions. Please answer the following questions. (25%)
   (1) List the keys that must be inserted into the hash table before 52. (12%)
   (2) List the key that must be the last one insertted into the hash table. (5%)
   (3) Calculate the number of insertion sequences that can result in the hash table as same as shown in Figure 2. (8%)

0	
1	
2	42
3	23
4	34
5	52
6	46
7	33
8	
9	

Figure 2. A hash table.

4. Use the stack data structure to answer the following questions. (20%)
   (1) Assume the stack ***s*** is initially empty. Illustrate the content of ***s*** after executing the following commands. (10%)
   s.push(16); s.push(15); s.push(29); s.push(24); s.push(19);

   (2) Update the content of the stack ***s*** in Question (1) by using the standard stack operations including **push**, **pop** and **empty** along with ***s*** and a temporary stack ***temp*** to get the results shown in Figure 3. (10%)

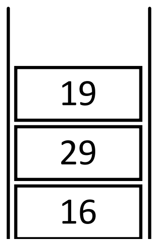


Figure 3. The stack ***s***.

5. Quick sort algorithm (25%)

(1) The C-language program in Figure 4 is aimed at implementing the **quick sort algorithm** for sorting eleven integers. Please write out the codes necessary for the blanks in Figure 4 to complete the program. Note: Write the code on your answer sheets with the corresponding number assigned to each blank. (13%)

```
typedef (1)   {
   int (2)   ;
} element;

void swap(element (3)   a, element (4)   b){
   element temp = (5)   ;
   *a = *b;
   (6)    = temp;
}
```

```
int main(){
   element es[11];   int values[11]= {12, 2, 16, 30,
8, 28, 4, 10, 20, 6, 18};
   for (int i = 0; i < 11; i++){
         es[i].key = values[i];
   }
   quickSort((7)   , (8)   , (9)   );
   return 0;
}
```

```
void quickSort((10)    a[], int left, int right){
      int reference, i, j;
      if (left < right){
            i = left; j = right + 1; reference = a[left];
            do {
                  do i++; (11)    (a[i].key < reference);
                  do j--; (12)    (a[j].key > reference);
                  if (i < j) swap(&a[i], &a[j]);
            } (13)    (i < j);
            swap(&a[left], &a[j]);
            quickSort(a, left, j - 1);
            quickSort(a, j + 1, right);
      }
}
```

Figure 4. Quick sort in C.

(2) According to the above implementation, the table in Figure 5 has illustrated the sequence of the eleven integers at the first two passes of the quickSort algorithm. Please write out the sequence of the eleven integers at each one of the remaining passes of the quickSort algorithm. (12%)

Pass	**ID (#)**	0	1	2	3	4	5	6	7	8	9	10
0	**Elements (key)**	12	2	16	30	8	28	4	10	20	6	18
1	**Elements (key)**	4	2	6	10	8	12	28	30	20	16	18
…												
13												

Figure 5. The passes of quick sort.